

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-69418

(43) 公開日 平成8年(1996)3月12日

(51) Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 12/12		F 7623-5B		
9/45				C4
9/44	5 5 4 Z	7737-5B		
		7737-5B		
			G 0 6 F 9/44	3 2 2 H C4
			15/16	3 2 0 K

審査請求 未請求 請求項の数7 O L (全 39 頁) 最終頁に続く

(21) 出願番号 特願平6-202565

(22) 出願日 平成6年(1994)8月26日

(71) 出願人 000002945

オムロン株式会社

京都府京都市右京区花園土堂町10番地

(72) 発明者 前田 匡

京都府京都市右京区花園土堂町10番地 オムロン株式会社内

(72) 発明者 十河 太治

京都府京都市右京区花園土堂町10番地 オムロン株式会社内

(72) 発明者 堤 康弘

京都府京都市右京区花園土堂町10番地 オムロン株式会社内

(74) 代理人 弁理士 古谷 栄男 (外2名)

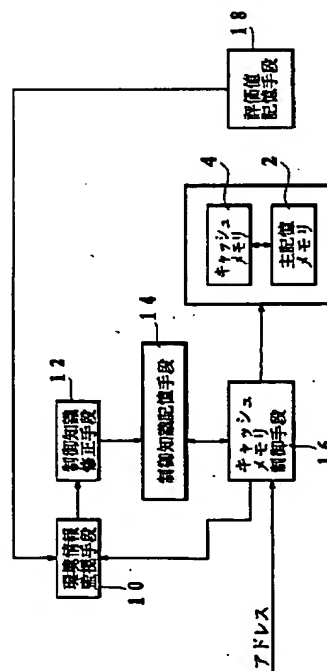
最終頁に続く

(54) 【発明の名称】 キャッシュメモリ制御システム、コンパイラ制御システムおよびコンピュータ・システム

(57) 【要約】

【目的】 コンピュータ装置およびその周辺装置の制御の高速化、最適化を目的とする。

【構成】 キャッシュメモリ制御手段16は、アドレスを受けて、当該アドレスがキャッシュメモリになれば、制御知識に基づいて、キャッシュメモリのアドレスの置き換えを行う。制御知識記憶手段14には、前記制御知識が記憶されている。環境情報監視手段10は、キャッシュメモリ制御手段16から、キャッシュメモリのアクセスの状態情報を得て、アクセス回数、キャッシュミス率を含む特徴量を算出する。さらに、キャッシュミス率が評価値よりも高いか否かを判断し、高い場合には、当該特徴量を制御知識修正手段12に出力するとともに制御知識修正手段12を動作させる。制御知識修正手段12は、前記特徴量に基づいて、制御知識記憶手段14に記憶された制御知識を修正する。



3

先度を決定し、
評価関数がタスク分散であるかタスク集中であるか、および評価値を記憶しておく、
前記環境情報を得て、タスク分散度またはタスク集中度を算出するとともに、評価関数がタスク分散またはタスク集中のいずれであるかを判断し、
評価関数がタスク分散である場合には、タスク分散度が評価値を下回っている場合に、評価関数がタスク集中である場合には、タスク集中度が評価値を上回っている場合に、前記環境情報に基づいて、各タスクの重要度、混雑度を算出し、算出したタスクの重要度、混雑度に基づいて、前記タスク制御知識を修正するとともに、前記タスクの重要度、混雑度に基づいて、前記評価関数および評価値を修正すること、
を特徴とするコンピュータの動作制御方法。

【 発明の詳細な説明】

【 0 0 0 1 】

【 産業上の利用分野】 この発明は、コンピュータ装置およびその周辺装置の制御に関し、特にその最適化に関するものである。

【 0 0 0 2 】

【 従来の技術】 図6 3 に、従来のキャッシュメモリ制御システムの構成を示す。容量が大きくアクセスの遅い主記憶メモリ2と容量は小さいがアクセスの早いキャッシュメモリ4が設けられている。CPU(図示せず) 等からアドレスが与えられると、主記憶メモリ2の当該アドレスにアクセスするとともに、当該アドレスを含む1群(データブロックという) のデータをキャッシュメモリ4にコピーする。以後、このアドレスが与えられると、キャッシュメモリ制御手段6は、キャッシュメモリ4にアクセスを行う。したがって、アクセススピードが向上する。

【 0 0 0 3 】 図6 4 に示すように、キャッシュメモリ4に、C1～C4の4つのデータブロックが設けられており、それぞれにB3、B6、B9、B14がコピーされているとする。なお、図中の「0」～「60」の数字はアドレスを示している。この状態において、アドレス「4」が与えられると、キャッシュメモリ制御手段6は、キャッシュメモリ4中の何れかのデータブロックを削除する。その後、アドレス「4」の属するデータブロック「B2」を削除したデータブロックの代りにコピーする。このようにして、キャッシュメモリ4の記憶内容は、次々と新しいデータブロックに置き換えられていく。

【 0 0 0 4 】 ここで、キャッシュメモリ4のどのデータブロックを削除するかは、キャッシュメモリ制御手段6が、図6 5 の手順によって決定していた。まず、LRUフラグを記憶するためのカウンタバッファを設ける。ここで、LRUフラグとは、各データブロックについて、最後に使用したのが現在から何回前であったかを現わす

4

フラグである。ここでは、「0」が現在使用中、「1」が1回前に使用、「2」が2回前に使用、「3」が3回以前に使用したことを現わすものとする。

【 0 0 0 5 】 キャッシュメモリが図6 4 の状態にあるとき、与えられたアドレスが「9」であれば、キャッシュメモリ制御手段6は、キャッシュメモリ4のデータブロックC1にアクセスする(図6 5 参照) 。この時、ブロックC1のLRUフラグを「0」にする。他のブロックC2～C4のLRUフラグは「3」のままである。

【 0 0 0 6 】 次に与えられたアドレスが「20」であれば、キャッシュメモリ4のデータブロックC4にアクセスする。この時、ブロックC4のLRUフラグを「0」にするるとともに、ブロックC1のブロックC1のLRUフラグを「1」に変える。他のブロックC2、C3のLRUフラグは「3」のままである。

【 0 0 0 7 】 その後、アドレスが「53」「32」「10」「21」と与えられると、各ブロックC1、C2、C3、C4のLRUフラグは、図に示すように「1」「2」「3」「0」となる。次に、アドレス「12」が

与えられたとする。この場合、キャッシュメモリ4にはこのアドレスがコピーされていないので(キャッシュミスという) 、主記憶メモリ2にアクセスする。これとともに、アドレス「12」を含むデータブロックB4をキャッシュメモリ4にコピーする。つまり、キャッシュメモリ4のいずれかのデータブロックを削除して、データブロックB4に置き換える、この際、キャッシュメモリ4の置き換えは、LRUフラグの最も大きいデータブロック、すなわち、最も古く使われたデータブロックが対象となる。したがって、この場合には、データブロックC3が置き換え対象となる。

【 0 0 0 8 】 このようにして置き換えられたキャッシュメモリ4の内容を示す。ブロックC3にデータブロックB4の内容がコピーされている。以上のようにして、キャッシュメモリ4の記憶内容は、次々と新しいデータブロックに置き換えられていく。

【 0 0 0 9 】 図6 7 に、従来のコンパイル装置の構成を示す。高級言語によるソースプログラムが与えられると、コンパイラ100は、これを実行形式のプログラムに変換する。この際、ソースプログラム中の変数に対して、メモリまたはレジスタを割り当てる。メモリよりもレジスタの方が高速であるから、出現頻度の高い変数に対して、レジスタを割り当てるようにしている(制御知識) 。

【 0 0 1 0 】 図6 8 に、従来のコンパイル装置におけるレジスタ割り当てのフローチャートを示す。まず、ステップS102において、ソースコードの対象行に変数が存在するか否かを判定する。存在しなければ、ステップS106に進む。存在する場合には、当該変数がすでに抽出されている変数か否かを判定する(ステップS103) 。抽出されている変数の場合には、当該変数の出現

50

7

【 0 0 2 3 】この発明は、上記のような問題点を解決して、状況に応じて適切なレジスタ割り当てを得ることのできるコンパイラ制御システムおよび変数割り当て方法を提供することを目的とする。

【 0 0 2 4 】また、図70、図71に示すようなコンピュータ・システムにおいては、次のような問題点があった。従来のシステムでは、各CPUの負荷の均等化を図って、特定のCPUに負担がかからないようにし、タスク全体の処理速度を維持するようにしている。しかし、全タスク量が少ない場合であっても、タスクの分散を図る10ように制御していた。このような場合には、タスクが分散されているため、ネットワークを介しての通信時間が大きくなり、かえって、処理速度が遅くなるという問題があった。

【 0 0 2 5 】この発明は、上記のような問題点を解決して、状況に応じて適切なタスク制御を行うことのできるシステムおよび方法を提供することを目的とする。

【 0 0 2 6 】

【課題を解決するための手段】請求項1、2のキャッシュメモリ制御システムは、アドレスを受けて、当該アドレスがキャッシュメモリになれば、制御知識に基づいて、キャッシュメモリのアドレスの置き換えを行うキャッシュメモリ制御手段、前記制御知識を記憶する制御知識記憶手段、キャッシュメモリ制御手段から、キャッシュメモリのアクセスの状態情報を得て、アクセス回数、キャッシュミス率を含む特徴量を算出し、キャッシュミス率が評価値よりも高いか否かを判断し、高い場合には、当該特徴量を制御知識修正手段に出力するとともに制御知識修正手段を動作させる環境情報監視手段、前記10特徴量に基づいて、制御知識記憶手段に記憶された制御知識を修正する制御知識修正手段、を備えている。

【 0 0 2 7 】ここで、環境情報監視手段は、実施例においては、図3、図4に示すフローチャートが対応する。また、制御知識修正手段は、実施例においては、図4のステップS12が対応する。なお、制御知識修正手段は、実施例では、図7のルール、図8のメンバーシップ関数に基づいて、ファジイ推論を行って制御知識の修正をしているが、環境情報監視手段からの特徴量に基づき、ファジイ推論に拘らず、制御知識の修正を行うものを含むものである。制御知識記憶手段に記憶される制御知識は、実施例では、図9、図10に示すものが対応する。なお、制御知識とは、キャッシュメモリのアドレス置き換えを行う際に、用いる規則、原則、優先度等をいうものである。キャッシュメモリ制御手段は、実施例においては、図11のフローチャートが対応する。

【 0 0 2 8 】請求項3のキャッシュメモリ制御方法は、キャッシュメモリへのアクセスにおけるキャッシュミス率を計測し、キャッシュミス率が所定条件よりも悪い場合には、前記制御知識を変更することを特徴としている。

8

【 0 0 2 9 】請求項4のコンパイラ制御システムは、ソース形式のプログラムを受けて、実行形式のプログラムを出力するとともに、制御知識に基づいて各変数をレジスタもしくはメモリに割り当てるコンパイラ制御手段、前記制御知識を記憶する制御知識記憶手段、コンパイラ制御手段から、変数情報およびループプログラム情報を得るとともに、コンパイラ評価手段からの評価値が所定の値よりも小さい場合には、前記変数情報およびループプログラム情報に基づいて変数出現度、ループプログラム出現度を含む特徴量を算出する環境情報監視手段、前記特徴量に基づいて、制御知識記憶手段に記憶された制御知識を修正する制御知識修正手段、を備えている。

【 0 0 3 0 】ここで、コンパイラ制御手段は、実施例においては、図15のステップS120が対応する。環境情報監視手段は、実施例においては、図15のステップS122以下、および図16のフローチャートが対応する。また、変数情報とは、特徴量を得るために必要な変数に関する情報をいうものであり、実施例では、変数の出現回数等がこれに対応する。環境変化理解手段は、実施例においては、図18のフローチャートが対応する。制御知識修正手段は、実施例においては、図23のフローチャートが対応する。制御知識修正手段は、実施例では、図24のルール、図25のメンバーシップ関数に基づいて、ファジイ推論を行って制御知識の修正をしているが、環境情報監視手段からの特徴量に基づき、ファジイ推論に拘らず、制御知識の修正を行うものを含むものである。

【 0 0 3 1 】請求項5のコンパイラにおける変数のレジスタへの割り当て制御方法は、変数をレジスタに割り当てるための制御知識を記憶しておき、コンパイル対象の高級言語ソースプログラムの中での各変数の出現状況を示す情報を抽出し、前記制御知識と出現状況に基づいて、レジスタに割り当てるべき変数を決定し、決定したレジスタ割り当てもとで、前記高級言語ソースプログラムをコンパイルした後に実行形式プログラムを生成し、前記実行形式プログラムの実行速度が所定の基準値よりも小さい場合には、前記制御知識を修正した後に、前記高級言語ソースプログラムを再度コンパイルすることを特徴としている。

【 0 0 3 2 】請求項6のコンピュータ・システムは、各CPUの負荷状況、主記憶装置の残り容量、既に実行されているタスクの処理状況、実行予定のタスクを含む環境情報を受けて、タスク制御知識に基づいて、各タスクをどのCPUで処理するか、および各タスクの優先度を決定するタスク制御手段、前記タスク制御知識を記憶するタスク制御知識記憶手段、評価関数がタスク分散であるかタスク集中であるかを記憶するとともに、評価値を記憶するタスク評価値記憶手段、タスク制御手段から、前記環境情報を得て、タスク分散度またはタスク集中度を算出するとともに、タスク評価値記憶手段に記憶され

11

ち、最新の1024回がレジスタに記憶される。

【0040】次に、ステップS3において、キャッシュミスが発生したか否かを判定する。キャッシュミスが発生していなければ、ステップS1に戻る。キャッシュミスが発生すれば、キャッシュミス記憶用のレジスタに、各ブロックC1～C4ごとに記録する(ステップS4)。このレジスタの記憶状態は、例えば、図2Bのようになる。図中、「1」の部分、アクセスミスがあったことを示している。

【0041】次に、リード・ライト(Read-Write)が発生したかどうかを判定する(ステップS5)。ここで、リード・ライト(Read-Write)とは、書込を行った場合を含む。リード・ライト(Read-Write)が発生していなければ、ステップS7に進む。発生していれば、図2と同様の構成のリード・ライト(Read-Write)記憶用レジスタに記憶する(ステップS6)。

【0042】次に、アクセス記憶用レジスタとキャッシュミス記憶用のレジスタに基づいて、キャッシュメモリ全体(全ブロック)のキャッシュミス率を演算する(ステップS7)。このキャッシュミス率が、評価値記憶手段18に記憶された評価値(この実施例では10%)より小さい場合には、ステップS1に戻る。つまり、制御知識を修正する必要はないと判断し、現在の制御知識のまま制御を進める。

【0043】一方、キャッシュミス率が、10%より大きい場合には、ステップS9に進む。ステップS9においては、各ブロックごとのキャッシュミス発生率を演算する。次に、各ブロックごとのリード・ライト(Read-Write)発生率、全体のリード・ライト(Read-Write)発生率を算出する(ステップS10、S11)。このようにして算出したキャッシュミス発生率、リード・ライト(Read-Write)発生率を、特徴量として制御知識修正手段12に渡す。例えば、各記憶用レジスタの集計内容が図5に示すような状態であった場合には、図6に示すような特徴量が出力される。

【0044】制御知識修正手段12は、この特徴量に基づいて制御知識の修正を行う(ステップS12)。この実施例では、制御情報として、LRU情報だけを用いるもの、LRU情報と使用頻度情報とを用いるもの、LRU情報とRead-Write情報とを用いるもの、LRU情報と使用頻度情報とRead-Write情報とを用いるものの4つの内から選ぶようにしている。

【0045】制御知識修正手段12は、特徴量に基づいて、図7のルールおよび図8のメンバーシップ関数を用いて、各ブロックごとにファジィ推論を行う。この実施例では、後件部の適合度が0.9を超えるものが1ブロックでも存在する場合には、当該後件部にしたがって、制御知識を変更(知識の追加、削除)する。例えば、現在、LRU情報だけを制御情報として用いているとし

12

て、「使用頻度情報を制御知識に加える」という後件部の適合度が0.9を超えた場合には、LRU情報と使用頻度情報とを用いるものに制御情報を変更する。

【0046】この実施例において用いた4つの制御情報を、図9、図10に示す。制御知識記憶手段14には、制御知識修正手段12によって選択された制御知識が記憶される。キャッシュメモリ制御手段16は、この制御知識に基づいて制御、すなわち、キャッシュミスが発生した場合にどのブロックを置き換えるかの制御を行う。なお、この実施例では、上記置き換え制御のために、図12に示すような、LRUフラグ、使用頻度フラグ、Read-Writeフラグを設けている。

【0047】LRUフラグは、各ブロックC1～C4に対して、今回アクセスした場合には「0」、前回アクセスした場合には「1」、前々回アクセスした場合には「2」、3回前にアクセスした場合には「3」が与えられる。図12Aは、LRUフラグの時間的変化を示している。使用頻度フラグは、図2Aのレジスタと同じものである。また、Read-Writeフラグは、図2Bのレジスタと同じものである。

【0048】キャッシュミスが発生すると、キャッシュメモリ制御手段16は、図11の置き換えブロック決定処理を行う。ここでは、図10Bの制御知識(LRU情報と使用頻度情報とRead-Write情報とを用いるもの)が用いられているものとして説明を行う。

【0049】まず、各ブロックC1～C4ごとに、LRUフラグの重み係数を決定する(ステップS20)。つまり、図10Bの制御知識に基づき、LRUフラグの値に対応した重み係数を算出する。この例でいえば、ブロックC1のLRUフラグは「3」であるから、重み係数は「1.0」となる(図13参照)。同様に、ブロックC2の重み係数は「0.5」、ブロックC3の重み係数は「0.75」、ブロックC4の重み係数は「0.25」となる。

【0050】次に、各ブロックC1～C4ごとに、使用頻度フラグの重み係数を決定する(ステップS21)。この重み係数を得るため、まず、図12Bに示すように、各ブロックごとのアクセス回数を計数する。このアクセス回数に基づき、図10Bの制御知識を使って、使用頻度フラグの重み係数を算出する。この例でいえば、ブロックC1のアクセス回数は他のどのブロックよりも多いので、重み係数は「0.25」となる(図13参照)。同様に、ブロックC2の重み係数は「0.75」、ブロックC3の重み係数は「0.5」、ブロックC4の重み係数は「0.5」となる。

【0051】次に、各ブロックC1～C4ごとに、Read-Writeフラグの重み係数を決定する(ステップS22)。この重み係数を得るため、まず、図12Cに示すように、各ブロックごとのRead-Write回数を計数する。このアクセス回数に基づき、図10B

を求める(ステップS156)。たとえば、図19Cに示すようになる。

【0064】次に、各サイクル時間における、各変数に対するレジスタの割り当て状況を取得する(ステップS158)。

【0065】制御知識修正手段106は、環境情報監視手段110と環境変化理解手段108からの出力に基づいて、制御知識の修正を行う。この実施例では、制御情報として、出現頻度のみを用いるもの、出現頻度と移動平均とを用いるもの、出現頻度とループプログラム出現頻度とを用いるもの、出現頻度と移動平均とループプログラム出現頻度とを用いるものの4つの中から選ぶようにしている。

【0066】制御知識修正手段106は、図23のフローチャートにしたがって、制御知識の修正を行う。まず、各変数ごとに、各サイクル時間ごとに、制御知識候補の推論を行う(ステップS160)。なお、推論は、ステップS150、154、156で求めた、出現場所(出現頻度がしきい値より大きいサイクル時間)についてのみ行う。この推論は、図24のルール、図25のメンバシップ関数に基づいて行う。後件部の適合度が0.9を超える制御知識候補がある場合には、当該後件部にしたがって、制御知識を変更する(ステップS164、166)。たとえば、現在、出現頻度だけを制御情報として用いているとして、「移動平均情報を制御知識に加える」という後件部の適合度が0.9を超えた場合には、出現頻度と移動平均とを用いるものに制御情報を変更する。

【0067】制御知識記憶手段104には、制御知識修正手段106によって選択された制御知識が記憶される。コンパイラ制御手段102は、この制御知識に基づいて制御、すなわちどの変数をレジスタに割り当てるのかの制御を行う。なお、この実施例では、上記割り当て制御のために、図26に示すような、出現頻度カウンタバッファ、移動平均バッファ、ループプログラム出現頻度カウンタを設けている。

【0068】出現頻度カウンタバッファは、コンパイル対象となっている当該サイクル時間における変数の出現頻度をカウントするものである。また、移動平均バッファは、コンパイル対象となっているサイクル時間、1つ前のサイクル時間および2つ前のサイクル時間の出現頻度を平均した値を保持するためのバッファである。ループプログラム出現頻度カウンタバッファは、コンパイル対象となっている当該サイクル時間におけるループプログラム中の変数の出現頻度をカウントするものである。

【0069】図27に、コンパイラ制御手段102のフローチャートを示す。まず、ソースプログラムの対象となるステップに変数が存在するか否かを判断する(ステップS170)。存在しなければ、ステップS174に進む。存在すれば、当該変数がすでに抽出されている変

数であるか否かを判断する(ステップS171)。すでに抽出されている変数であれば、当該変数の出現頻度カウンタをインクリメントする(ステップS172)。抽出されていない変数であれば、当該変数のための出現頻度カウンタを用意するとともに、カウント数を「1」とする(ステップS173)。

【0070】次に、ステップS174において、ループプログラムが存在するか否かを判定する。存在しなければ、ステップS180に進む。存在すれば、当該ループプログラムの中に変数が存在するか否かを判断する(ステップS175)。存在しなければ、ステップS180に進む。存在すれば、当該変数がすでに抽出されている変数であるか否かを判断する(ステップS176)。すでに抽出されている変数であれば、当該変数のループプログラム出現頻度カウンタをインクリメントする(ステップS178)。抽出されていない変数であれば、当該変数のための出現頻度カウンタを用意するとともに、カウント数を「1」とする(ステップS179)。

【0071】次に、ステップS180において、所定ステップ数のコンパイルが進んだかどうか(サイクル時間が経過したかどうか)を判定する。経過していなければ、ステップS170以下を繰り返し実行する。経過していれば、レジスタの割り当てを決定する(ステップS181)。

【0072】レジスタの割り当て決定のフローチャートを、図30、図31に示す。なお、ここでは、図22Bに示す制御知識を用いて割り当ての決定を行うものとして説明を行う。まず、各変数ごとに、出現頻度がしきい値 α (この場合は0である(図22B参照))より大きな変数に対し、係数を決定する(ステップS186)。例えば、図32に示すように、各変数val1, val2, val3の出現頻度カウンタの値が、「100」「110」「120」であったとすれば、それぞれの重み係数は、図22Bより「0.5」「0.75」「1.0」と決定される。次に、同様に、ループプログラム出現頻度カウンタに基づき、その重み係数を決定する(ステップS188)。

【0073】次に、1回前のサイクル時間の頻度カウンタの値、2回前のサイクル時間の頻度カウンタの値、現在のサイクル時間の頻度カウンタの値に基づいて、各変数ごとに、移動平均を算出する(ステップS188)。この移動平均に基づいて、図22Bより、重み係数を算出する(ステップS189)。

【0074】このようにして、各変数ごとに3つの重み係数を算出した後、この重み係数を各変数ごとに合計して、評価値とする(ステップS190)。次に、この評価値の上位3つまでの変数を選定する(ステップS191)。ここで、レジスタの数よりも選定された変数の数の方が多いか否かを判定する(ステップS192)。少ない場合には、選定された変数をれじすたに割り当てる

スク集中度がタスク評価値記憶手段214に記憶された評価値(ここでは20%以下とする)以下であるか否かを判断する。ここで、各CPUごとのタスク集中度が全て20%以下である場合のみ、タスク集中度が20%以下であるとする(図40参照)。

【0085】タスク集中度が20%以下であれば、ステップS310に戻る。タスク集中度が20%以下でなければ、ステップS322以下を実行し、環境変理解手段220を動作させる。

【0086】以上のように、環境情報監視手段222は、環境情報から算出されたタスク分散度またはタスク集中度が、記憶された評価値に適合していない場合に、環境変理解手段220を動作させる。

【0087】図41に、環境変理解手段220のフローチャートを示す。環境変理解手段220は、環境情報監視手段222からの出力を受けて動作する。まず、ステップS330において、各タスクの重要度を算出する。この実施例では、各タスクの発生頻度、各タスクの処理平均時間に基づいて、タスクの重要度をファジイ推論する。つまり、図49A、Bの前件部のメンバーシップ関数、図48のルール、図49Cの後件部のメンバーシップ関数に基づいて、タスクの重要度を求める。その結果を、図50に示すタスクの重要度バッファに記憶する(ステップS331)。

【0088】次に、分散コンピューティング環境の混雑度を、各CPUの負荷、全体のタスク数に基づいて、ファジイ推論する(ステップS332)。つまり、図52A、Bの前件部のメンバーシップ関数、図51のルール、図52Cの後件部のメンバーシップ関数に基づいて、分散コンピューティング環境の混雑度を求める。ここで、分散コンピューティング環境の混雑度とは、CPUの負荷や、ネットワークの混み具合によって表される、システム全体の混雑度合いをいう。タスク重要度および分散コンピューティング環境の混雑度は、環境情報評価手段216および制御知識修正手段218に出力される。

【0089】次に、図42に、環境情報評価手段216のフローチャートを示す。まず、ステップS340において、タスクの頻度、分散コンピューティング環境の混雑度、タスク重要度に基づいて、評価関数をタスク分散とするか、タスク集中とするかを推論する。すなわち、図54A、B、Cの前件部のメンバーシップ関数、図53のルール、図54Dの後件部のメンバーシップ関数に基づいて、評価関数をタスク分散とするか、タスク集中とするかを各タスク毎に推論する。その結果、タスク分散とすべき推論値とタスク集中とすべき推論値の何れが大きいかを判断する(ステップS341)。

【0090】タスク集中の方が大きい場合には、評価関数にタスク集中を採用することとし(ステップS342)、当該評価関数および評価値をタスク評価値記憶手

段214に書き込む(ステップS343)。タスク集中の場合の、評価関数および評価値の一例を、図43Bに示す。

【0091】一方、タスク分散の方が大きい場合には、評価関数にタスク分散を採用することとし(ステップS344)、当該評価関数および評価値をタスク評価値記憶手段214に書き込む(ステップS345)。タスク分散の場合の、評価関数および評価値の一例を、図43Aに示す。

【0092】このようにして、環境に応じて、適切な評価関数、評価値が選択される。

【0093】また、環境変理解手段220からの出力は、制御知識修正手段218にも与えられている。図44に、制御知識修正手段218のフローチャートを示す。まず、ステップS350において、各タスク毎に、分散コンピューティング環境の混雑度、タスクの頻度、タスクの重要度、タスクの優先度に基づいて、どの制御知識を用いるかをファジイ推論する。すなわち、図56A、B、C、Dの前件部のメンバーシップ関数、図55のルール、図56Eの後件部のメンバーシップ関数に基づいて、その制御知識を用いるかを各タスク毎に推論する。

【0094】次に、これら推論結果中に、適合度0.9以上のものがあるか否かを判断する(ステップS351)。あれば、当該適合度0.9以上の知識をタスク制御知識記憶手段212に書き込む。なお、適合度0.9以上のものが複数ある場合には、それら全ての知識を書き込む(ステップS352)。適合度0.9以上のものがなければ、知識の修正は行わない。

【0095】図45、図46に、タスク制御知識の例を示す。図45において、例えば、タスクを他のCPUに分散してもよい場合には、全てのCPUの処理許可フラグが「1」とされる。タスクを自分のマシンのCPUのみで行う場合には、タスク要求が発生したマシンのCPUの処理許可フラグが「1」とされ、他のマシンのCPUの処理許可フラグが「0」とされる。何れの知識を採用するかは、上述の制御知識修正手段218によって決定される(図56E参照)。

【0096】図46には、タスク優先順位を決定するための制御知識の例が示されている。図46Aが処理時間のみで優先順位を決定する知識であり、図46Bが使用頻度も考慮して優先順位を決定する知識である。何れの知識を採用するかは、上述の制御知識修正手段218によって決定される(図56E参照)。

【0097】最後に、タスク制御手段210の動作を説明する。まず、タスク実行先決定のフローチャートを図59に示す。まず、タスク実行先決定用フラグを読み込む(ステップS360)。つまり、制御知識によって決定された各CPUの処理許可フラグを読み込む。次に、ステップS361において、フラグが「0」であるCP

10

20

30

40

50

【図27】コンパイラ制御手段102の動作を示すフローチャートである。

【図28】コンパイラ制御手段102の動作を示すフローチャートである。

【図29】コンパイラ制御手段102の動作を示すフローチャートである。

【図30】コンパイラ制御手段102の動作を示すフローチャートである。

【図31】コンパイラ制御手段102の動作を示すフローチャートである。

【図32】コンパイラ制御手段102のレジスタ割り当てを示す図である。

【図33】実施例におけるレジスタ割り当ての状態を示す図である。

【図34】この発明の一実施例によるコンピュータ・システムを示す図である。

【図35】環境情報監視手段222の動作を示すフローチャートである。

【図36】環境情報監視手段222の動作を示すフローチャートである。

【図37】環境情報監視手段222の動作を示すフローチャートである。

【図38】環境情報監視手段222の動作を示すフローチャートである。

【図39】CPUの負荷レジスタを示す図である。

【図40】CPU負荷バッファを示す図である。

【図41】環境変化理解手段220の動作を示すフローチャートである。

【図42】環境情報評価手段216の動作を示すフローチャートである。

【図43】評価知識フラグと評価目標値を示す図である。

【図44】制御知識修正手段218の動作を示すフローチャートである。

【図45】タスク実行先決定用フラグの例を示す図である。

【図46】タスク優先順位用重み係数の例を示す図である。

【図47】環境情報の一例を示す図である。

【図48】各タスクの重要度を決定するためのルールを示す図である。

【図49】各タスクの重要度を決定するためのメンバーシップ関数を示す図である。

【図50】タスクの重要度バッファを示す図である。

【図51】分散コンピューティング環境の混雑度を決定するためのルールを示す図である。

【図52】分散コンピューティング環境の混雑度を決定するためのメンバーシップ関数を示す図である。

【図53】評価関数を決定するためのルールを示す図である。

【図54】評価関数を決定するためのメンバーシップ関数を示す図である。

【図55】制御知識を決定するためのルールを示す図である。

【図56】制御知識を決定するためのメンバーシップ関数を示す図である。

【図57】環境情報および使用頻度レジスタを示す図である。

【図58】タスク実行先決定用フラグ、タスク優先順位用重み係数を示す図である。

【図59】タスク実行先決定制御のフローチャートを示す図である。

【図60】タスク実行先決定用フラグを示す図である。

【図61】優先度決定処理のフローチャートを示す図である。

【図62】優先度決定処理を説明するための図である。

【図63】従来のキャッシュメモリ制御システムを示す図である。

【図64】従来のキャッシュメモリの割り当てを示す図である。

【図65】LRUフラグを示す図である。

【図66】キャッシュメモリに割り当てられたブロックを示す図である。

【図67】従来のコンパイラシステムを示す図である。

【図68】従来のコンパイラシステムの動作を示すフローチャートである。

【図69】従来のコンパイラシステムにおけるレジスタ割り当て状況を示す図である。

【図70】従来のコンピュータ・システムのブロック図を示す図である。

【図71】従来のコンピュータ・システムのブロック図を示す図である。

【図72】従来のタスク制御処理を示す図である。

【図73】従来のタスク制御知識を示す図である。

【符号の説明】

10・・・環境情報監視手段

12・・・制御知識修正手段

14・・・制御知識記憶手段

16・・・キャッシュメモリ制御手段

102・・・コンパイラ制御手段

104・・・制御知識記憶手段

106・・・制御知識修正手段

110・・・環境情報監視手段

210・・・タスク制御手段

212・・・タスク制御知識記憶手段

214・・・タスク評価値記憶手段

216・・・環境情報評価手段

218・・・制御知識修正手段

220・・・環境変化理解手段

222・・・環境情報監視手段

【 図2 】

A

ブロックC1	1	1	0	0	1	1	0	0	0	1	1	0	1	1	1	1	0
ブロックC2	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	1
ブロックC3	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
ブロックC4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

2回目 今回

B

ブロックC1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
ブロックC2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
ブロックC3	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
ブロックC4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

2回目 今回

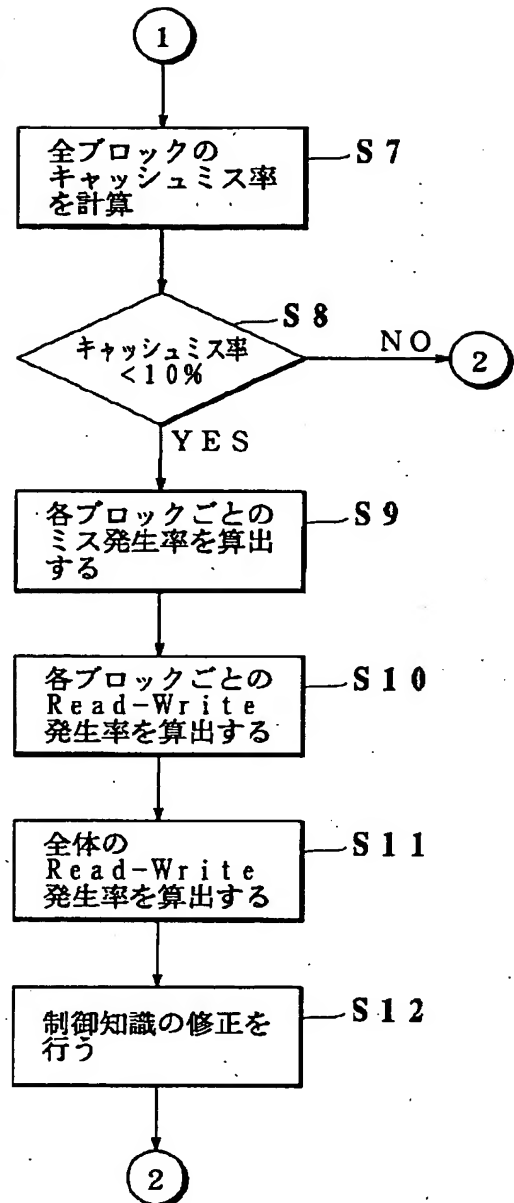
【 図5 】

ブロック 番号	最近1024回の キャッシュメモリ アクセス回数	最近1024回の キャッシュメモリ アクセス時の キャッシュミス 発生回数	最近1024回の キャッシュメモリ アクセス時の Read-Write 回数
1	512	71	15
2	256	85	30
3	192	19	0
4	64	6	1
合計	1024	131	36

【 図6 】

ブロック 番号	最近1024回の キャッシュメモリ アクセス回数	最近1024回の キャッシュメモリ アクセス時の キャッシュミス 発生率	最近1024回の キャッシュメモリ アクセス時の Read-Write 発生率
1	512	14%	3%
2	256	14%	12%
3	192	10%	0%
4	64	9%	2%
合計	1024	13%	4%

【 図4 】



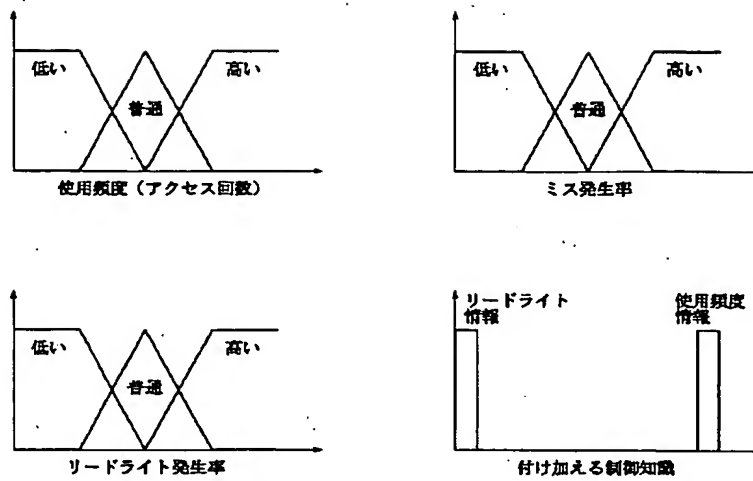
【 図7 】

IF
使用頻度が高い、かつ、ミス発生率が高い
Then
使用頻度情報を制御知識に加える

IF
ミス発生率が高い、かつ、リード・ライト発生率が高い
Then
リード・ライト情報を制御知識に加える

...

【 図8 】



【 図9 】

A

・LRU情報だけを用いる

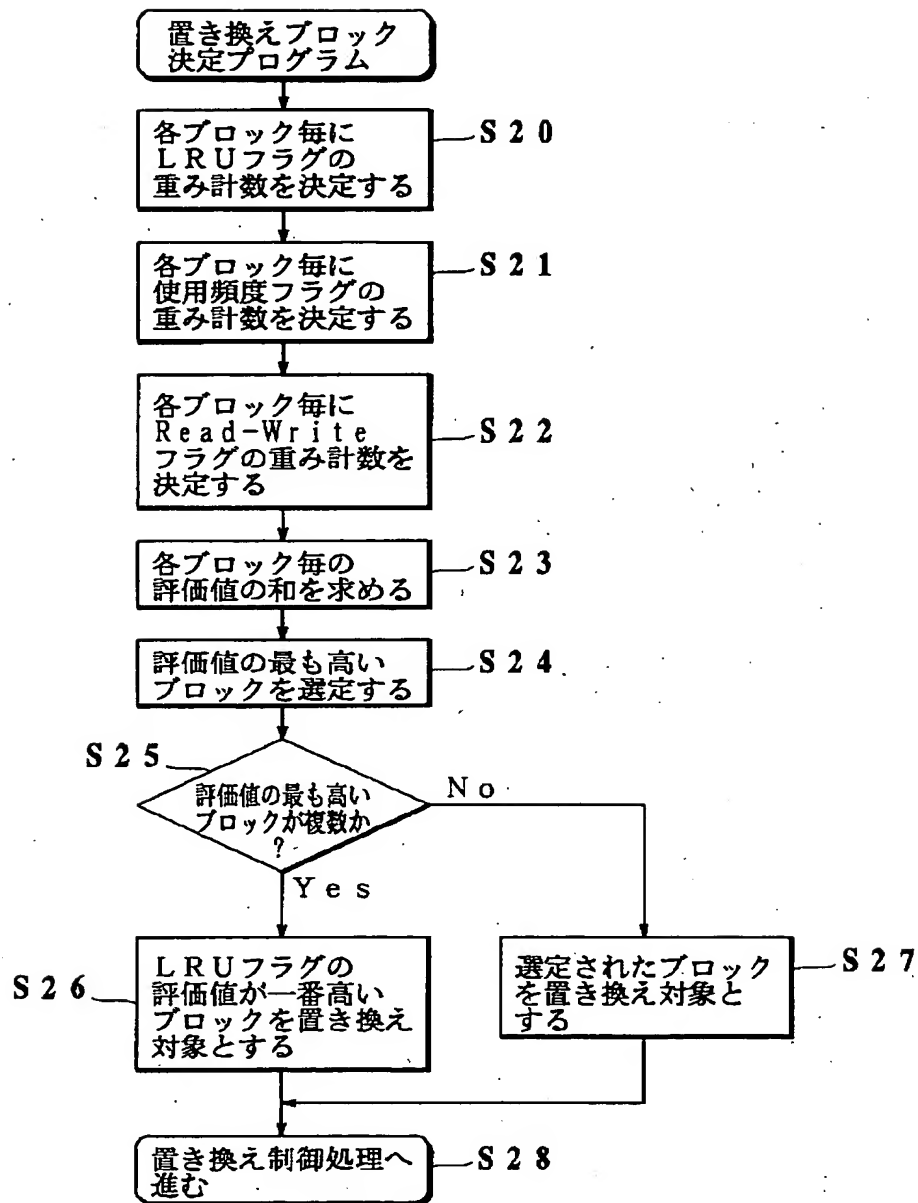
LRU フラグ	現在使用中	1つ前に使用	2つ前に使用	3つ前に使用
使用頻度 フラグ	0	0	0	1. 0
Read-Write フラグ	Read-Write 回数1位	Read-Write 回数2位	Read-Write 回数3位	Read-Write 回数4位
	0	0	0	0

B

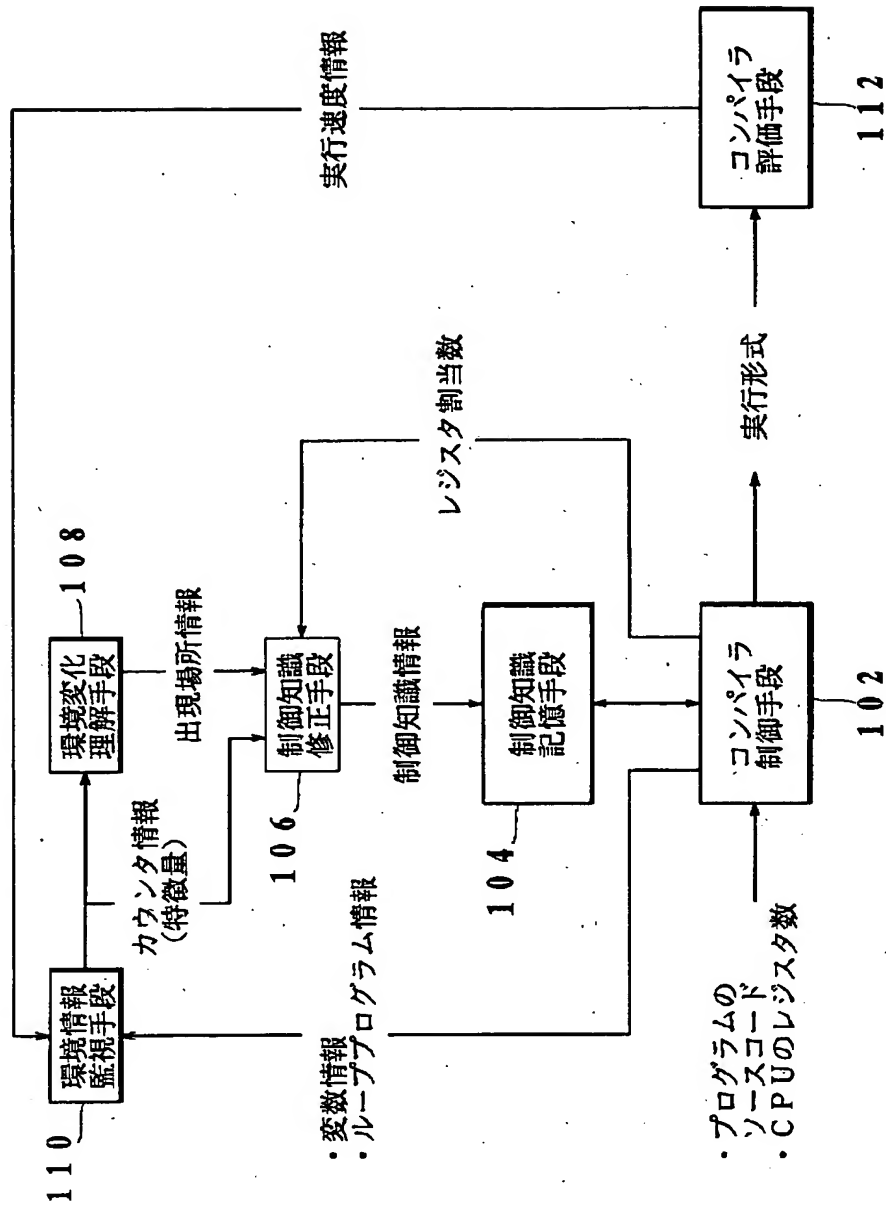
・LRU情報と使用頻度とを用いる

LRU フラグ	現在使用中	1つ前に使用	2つ前に使用	3つ前に使用
使用頻度 フラグ	0. 25	0. 5	0. 75	1. 0
Read-Write フラグ	Read-Write 回数1位	Read-Write 回数2位	Read-Write 回数3位	Read-Write 回数4位
	0	0	0	0

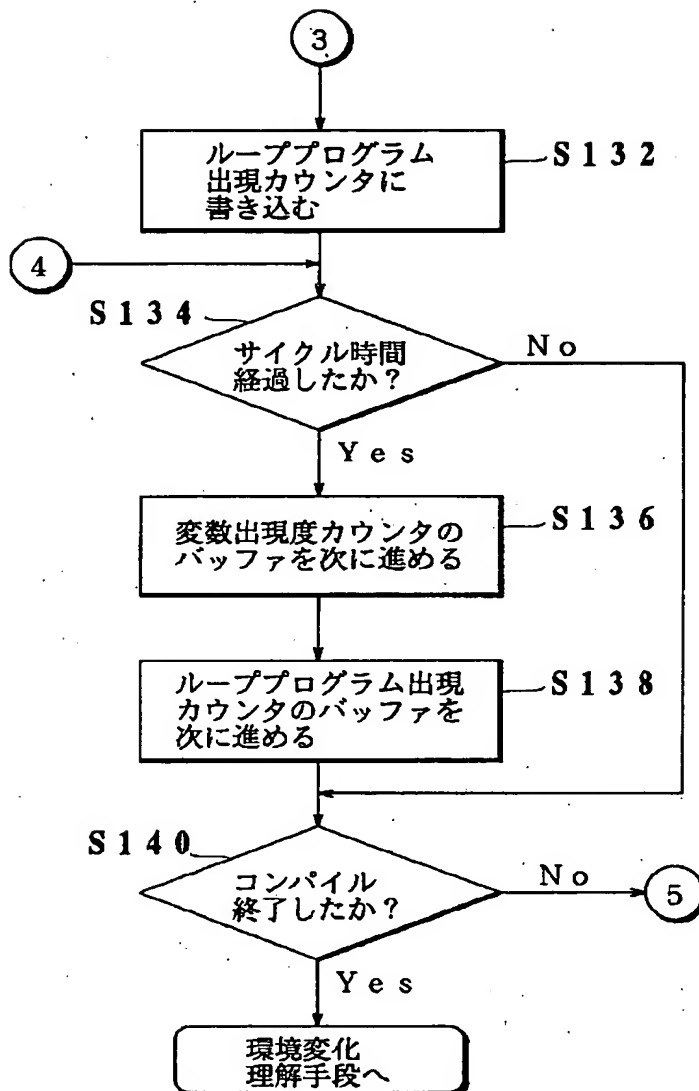
【 図1 1 】



【 図1 4 】



【 図1 6 】



【 図4 5 】

タスク制御知識

タスク実行先決定用フラグの例

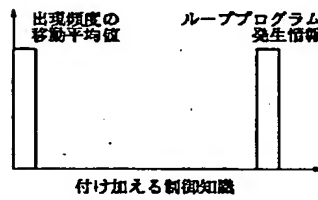
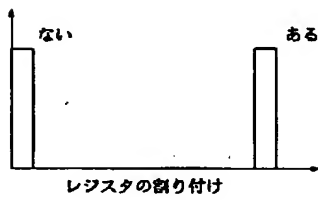
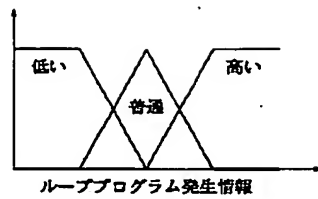
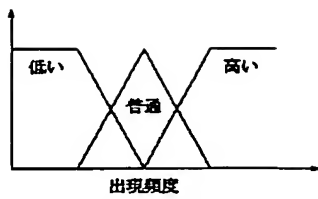
- ・タスクを分散しても良い場合

処理許可フラグ	すべてのCPUが '1'
---------	--------------

- ・タスクを自分のマシンのCPUで行なう場合

処理許可フラグ	タスク要求が発生したマシンのCPUのフラグは1
処理許可フラグ	タスク要求が発生していないマシンのCPUのフラグは0

【 図2 5 】

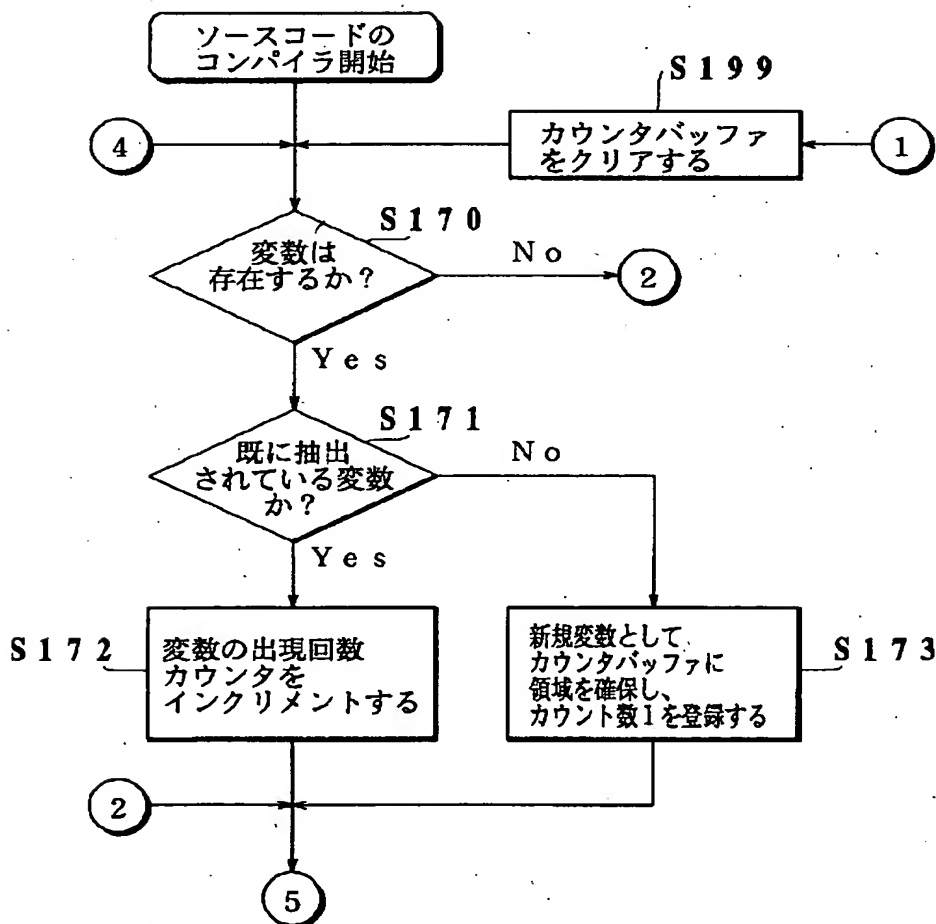


【 図5 1 】

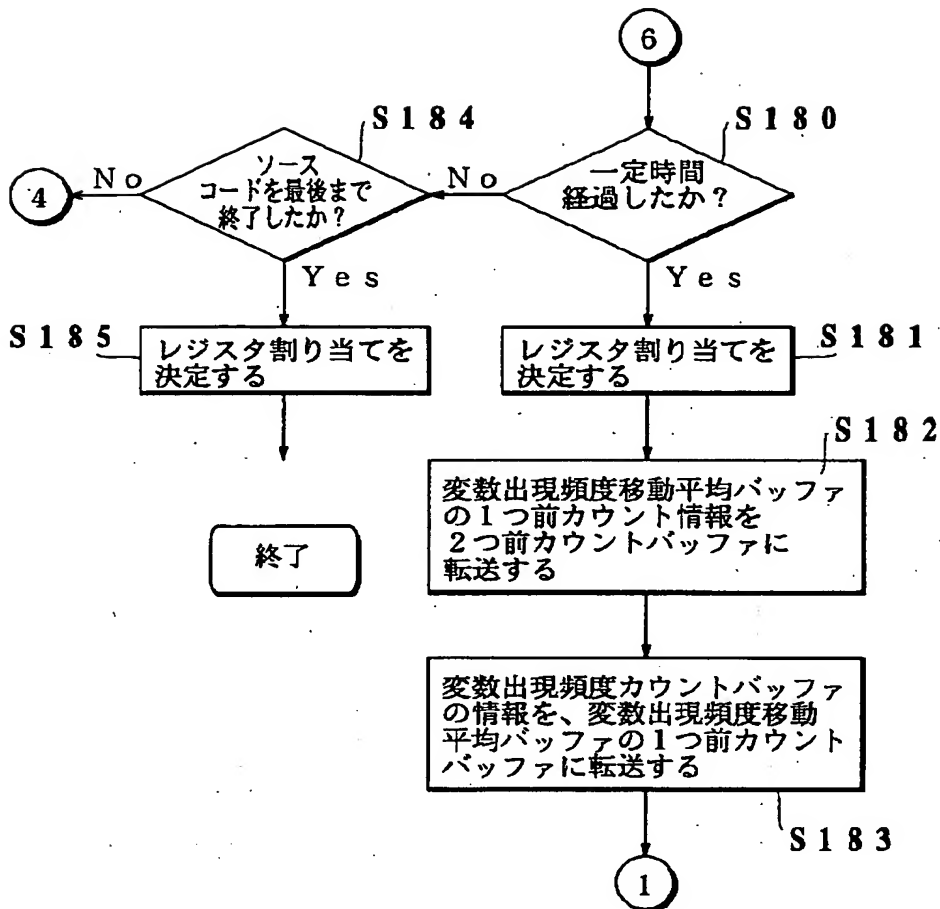
分散コンピューティング環境の混雑度
(CPUの負荷や、ネットワークの混み具合)

IF
CPU1の負荷が軽い、かつ、CPU2の負荷が軽い、かつ、
CPU3の負荷が軽い、かつ、全体のタスクが少ない
THEN
分散コンピューティング環境の混雑度は低い

【 図2 7 】



【 図29 】



【 図32 】

変数名	変数出現頻度 カウンタバッファ	重み係数
val 1	<input type="text"/>	0. 5
val 2	<input type="text"/>	0. 75
val 3	<input type="text"/>	1. 0

変数名	変数出現頻度 移動平均バッファ	重み係数
val 1	<input type="text"/>	0. 75
val 2	<input type="text"/>	1. 0
val 3	<input type="text"/>	0. 5

変数名	ループプログラム 出現カウンタ	重み係数
val 1	<input type="text"/>	1. 0
val 2	<input type="text"/>	0. 75
val 3	<input type="text"/>	0. 5

【 図55 】

IF
タスクの重要度が高い、かつ、
タスクの優先度が低い

THEN
制御知識に '使用頻度が多い時には優先度を上げる' 知識を
付け加える

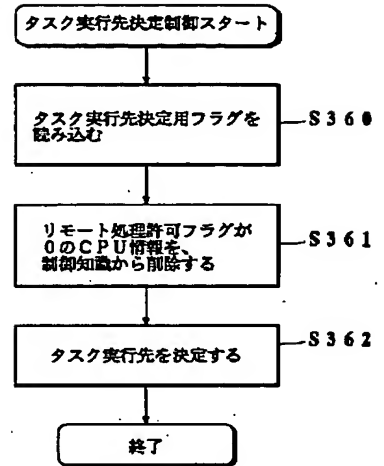
IF
分散コンピューティング環境の複雑度が低い、かつ、
存在するタスクは頻度が高い、かつ、
存在するタスクは重要度が低い

THEN
制御知識を 'タスクを自分のマシンのCPUで行なう' 知識に
置き換える

各変数の評価値の合計

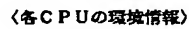
変数名	評価値
val 1	2. 25
val 2	2. 5
val 3	2. 0

【 図5 9 】



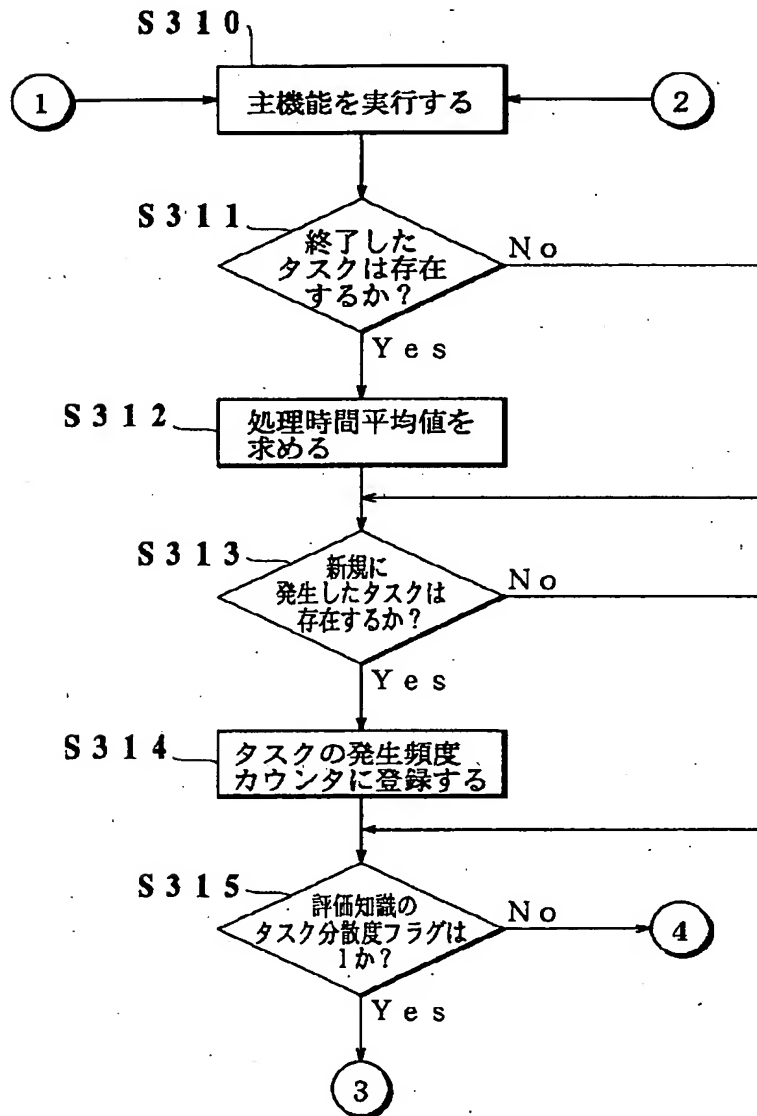
【 図5 7 】

〈各マシン1の取替情報〉

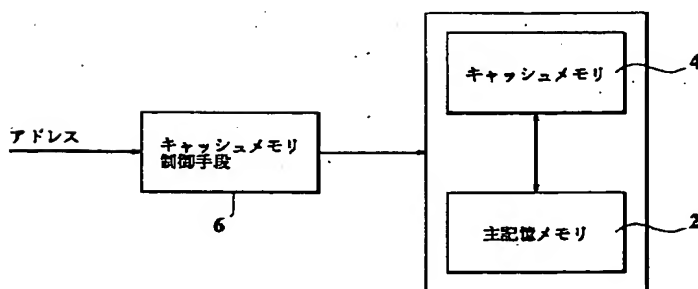
[illegible]

【 図3 5 】

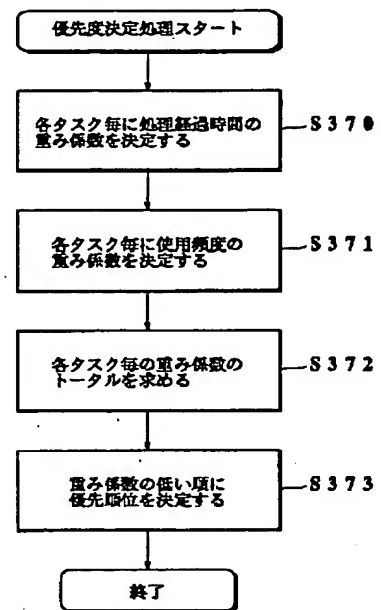
環境情報監視手段のフローチャート（その1）



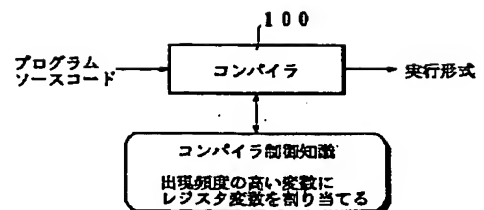
【 図6 3 】



【 図6 1 】

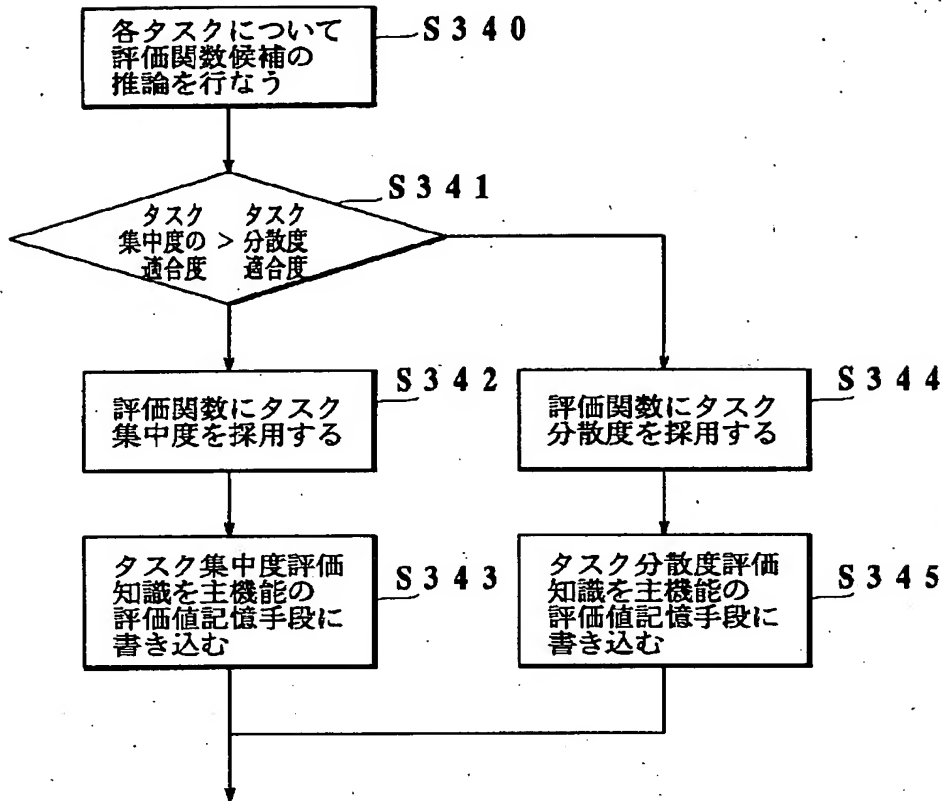


【 図6 7 】



【 図4 2 】

環境情報評価手段のフローチャート



【 図4 7 】

A 発生したタスクの
処理時間平均値

task 1	task 2	task 3	task 4	task 5	task 6	task 7
1'30"	0'50"	0'35"	4'12"	3'00"	2'30"	5'00"

B

タスクの発生頻度

トータル

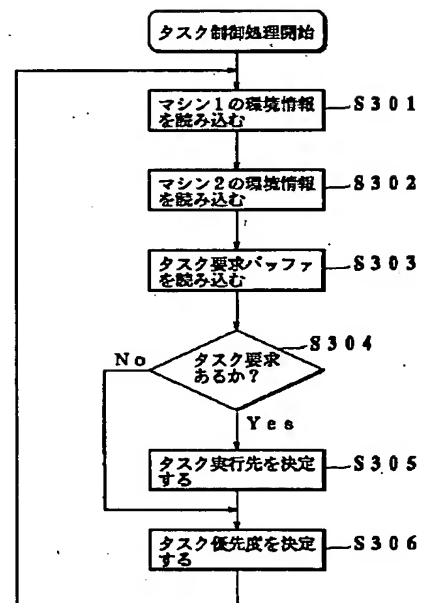
task 1	2	2	0	0	0	1	1	3	0	0	0	0	0	2	3	1	15
task 2	1	2	0	0	0	0	1	0	1	1	0	0	0	1	2	0	9
task 7	0	1	7	6	7	1	0	2	1	0	1	0	5	0	0	1	32

タスク カウント サイクル

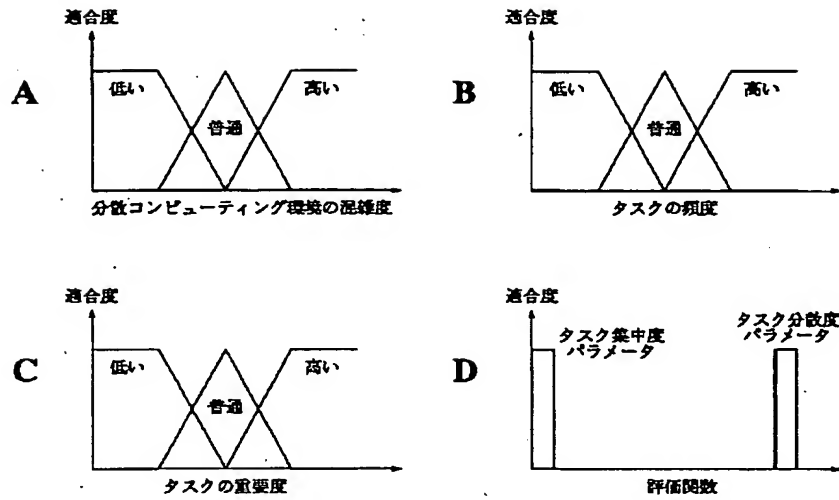
C

	CPU 負荷	主記憶の 残り容量
CPU1	70%	1.9M
CPU2	20%	12M
CPU3	40%	6M

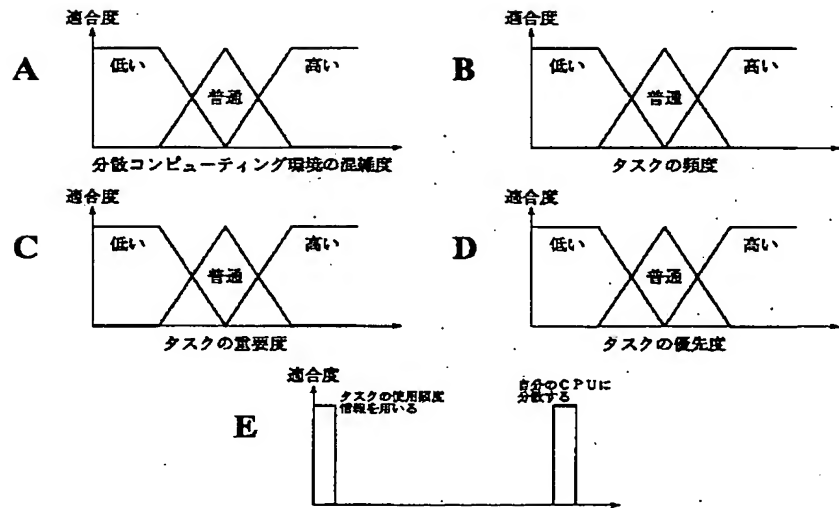
【 図7 2 】



【 図5 4 】



【 図5 6 】



【 図6 0 】

(タスク実行先決定用フラグ)

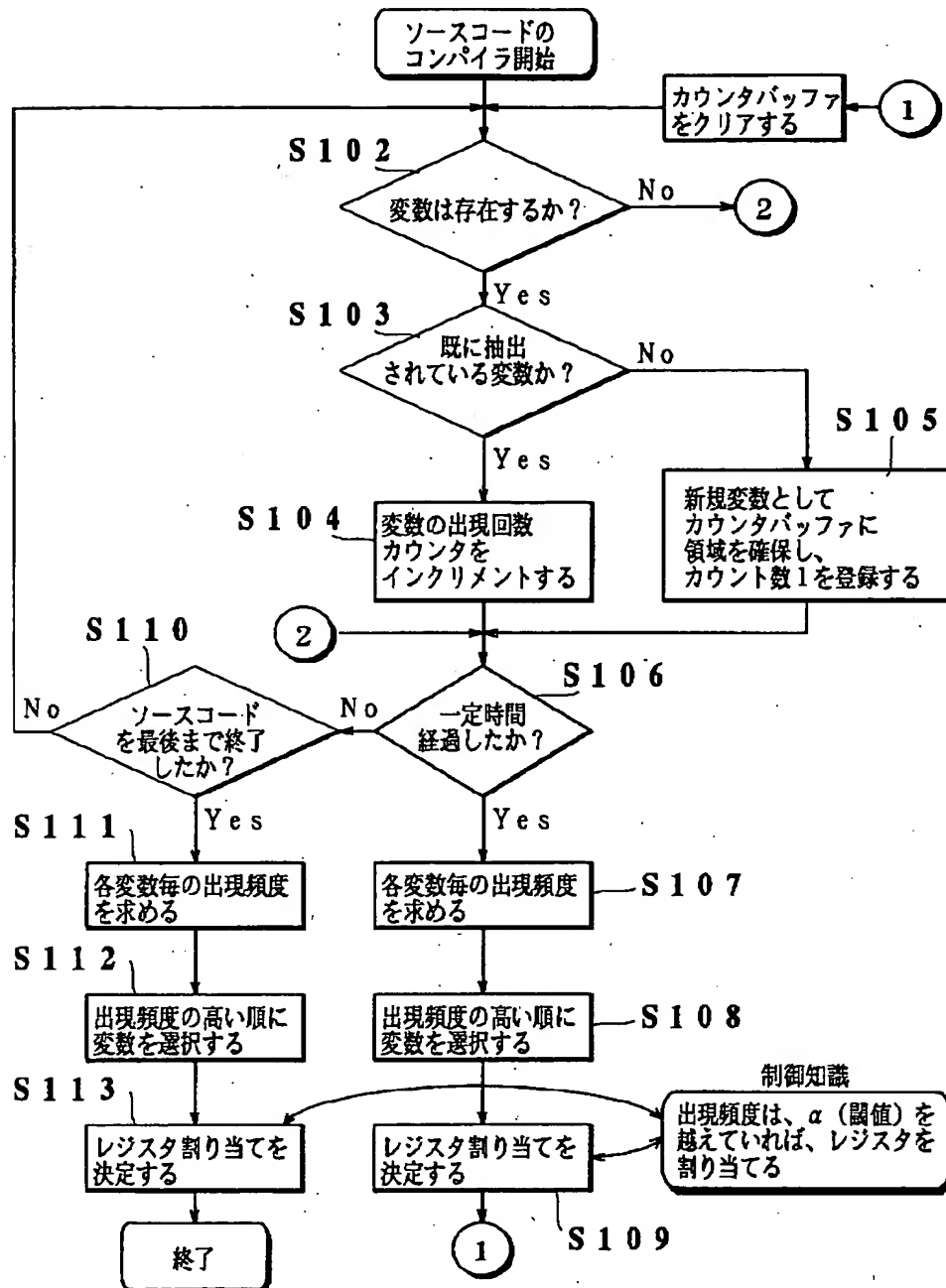
処理許可フラグ	1 (CPU1, CPU2), 0 (CPU3)			
---------	--------------------------	--	--	--

CPU3の情報は使用しない

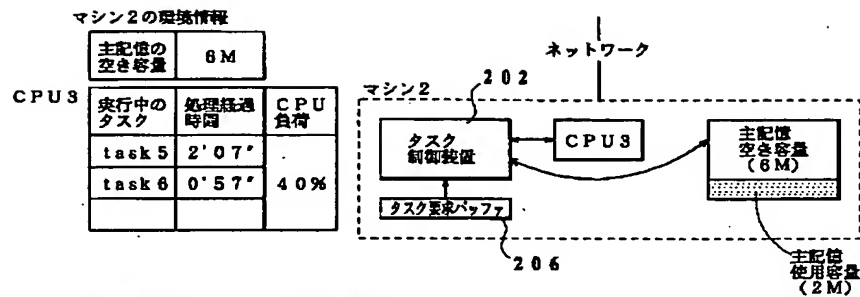
制御知識

	【条件A】 CPU負荷 最も軽い	【条件B】 主記憶の空き 容量最も大きい	【条件C】 【A】と【B】 の論理積	実行先CPUフラグ (【A】と【C】 の論理和)
CPU1	0	0	0	0
CPU2	1	1	1	1
CPU3	0	0	0	0

【 図6 8 】



【 図 7 1 】



タスク要求バッファ206の内容(例)

実行予定 タスク	タスク要求発生元	タスク名
	マシン2	task 7

【 図 7 3 】

タスク実行先決定制御知識

	タスク実行先決定制御知識			実行先CPUフラグ ([A]と[C] の論理和)
	[条件A] CPU負荷 最も軽い	[条件B] 主記憶の空き 容量最も大きい	[条件C] [A]と[B] の論理積	
CPU1	0	0	0	0
CPU2	1	0	0	1
CPU3	0	1	0	0

A

優先度決定知識…処理時間の長いタスクの優先度をあげる

B

タスク名	処理経過時間	タスク優先度
task 1	1' 12"	3
task 2	0' 32"	5
task 3	0' 15"	6
task 4	3' 12"	1
task 5	2' 07"	2
task 6	0' 57"	4

フロント ページの続き

(51) Int.Cl.⁶

G 0 6 F 9/46

15/163

識別記号 庁内整理番号

3 4 0 D 7737-5B

F I

技術表示箇所

(72)発明者 田島 年浩

京都府京都市右京区花園土堂町10番地 オムロン株式会社内

(72)発明者 荒尾 真樹

京都府京都市右京区花園土堂町10番地 オムロン株式会社内

(72)発明者 大八木 雅之

京都府京都市右京区花園土堂町10番地 オムロン株式会社内